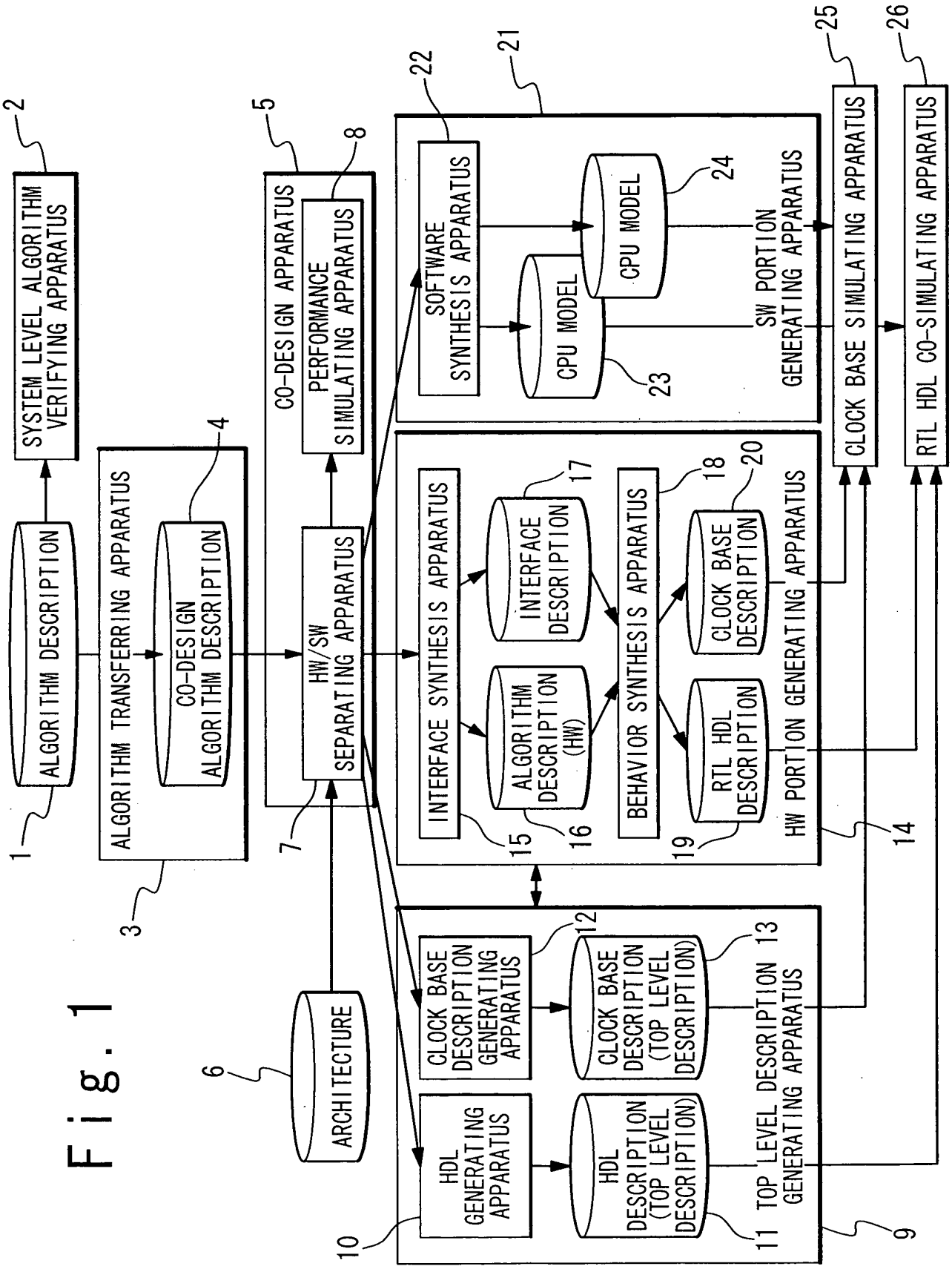

SYSTEM LEVEL ALGORITHM VERIFYING APPARATUS
ALGORITHM DESCRIPTION
ALGORITHM TRANSFERRING APPARATUS
CO-DESIGN ALGORITHM DESCRIPTION
CO-DESIGN APPARATUS
PERFORMANCE SIMULATING APPARATUS
HW/SW SEPARATING APPARATUS
SOFTWARE SYNTHESIS APPARATUS
CPU MODEL
CPU MODEL
SW PORTION GENERATING APPARATUS
INTERFACE SYNTHESIS APPARATUS
INTERFACE DESCRIPTION
ALGORITHM DESCRIPTION (HW)
BEHAVIOR SYNTHESIS APPARATUS
CLOCK BASE DESCRIPTION
RTL HDL DESCRIPTION
HW PORTION GENERATING APPARATUS
CLOCK BASE SIMULATING APPARATUS
RTL HDL CO-SIMULATING APPARATUS
ARCHITECTURE
CLOCK BASE DESCRIPTION GENERATING APPARATUS
CLOCK BASE DESCRIPTION (TOP LEVEL DESCRIPTION)
HDL GENERATING APPARATUS
HDL DESCRIPTION (TOP LEVEL DESCRIPTION)
TOP LEVEL DESCRIPTION GENERATING APPARATUS
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Fig. 1

Fig. 2A

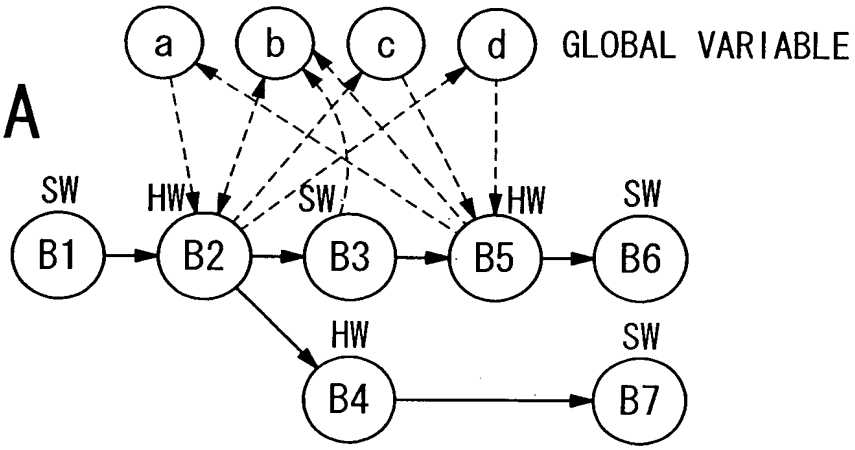

a
b
c
d
GLOBAL VARIABLE
SW
HW
SW
HW
SW
B1
B2
B3
B5
B6
HW
SW
B4
B7

Fig. 2B

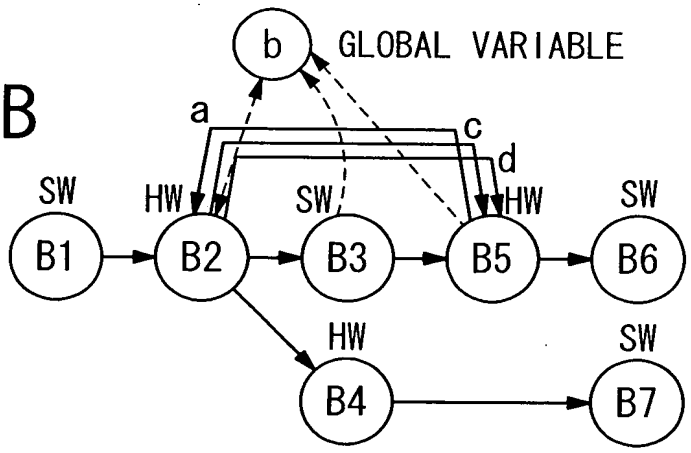

b
GLOBAL VARIABLE
a
c
d
SW
HW
SW
HW
SW
B1
B2
B3
B5
B6
HW
SW
B4
B7

Fig. 3A

```
int a, b, c, d;

funcB2(in1, *out1)  {
  b = a+b+in1;
  c = a+1;
  d = a+2;
  out1 = c+a;
}
funcB3(in2)  {
  b = in2+b;
}

funcB5(in3)  {
  a = in3+c+d;
  b = c+d
}
```

Fig. 3B

```
a:read
b:read, write
c:write
d:write

b:read, write

a:write
b:write
c:read
d:read
```

Fig. 3C

```
int b;

funcB2(in1, a, *out1, *c, *d)  {
  b=a+b+in1;
  c=a+1;
  d=a+2;
  out1=c+a;
}
funcB3(in2)  {
  b=in2+b;
}

funcB5(in3, *a, c, d)  {
  a=in3+c+d;
  b=c+d;
}
```

Fig. 4

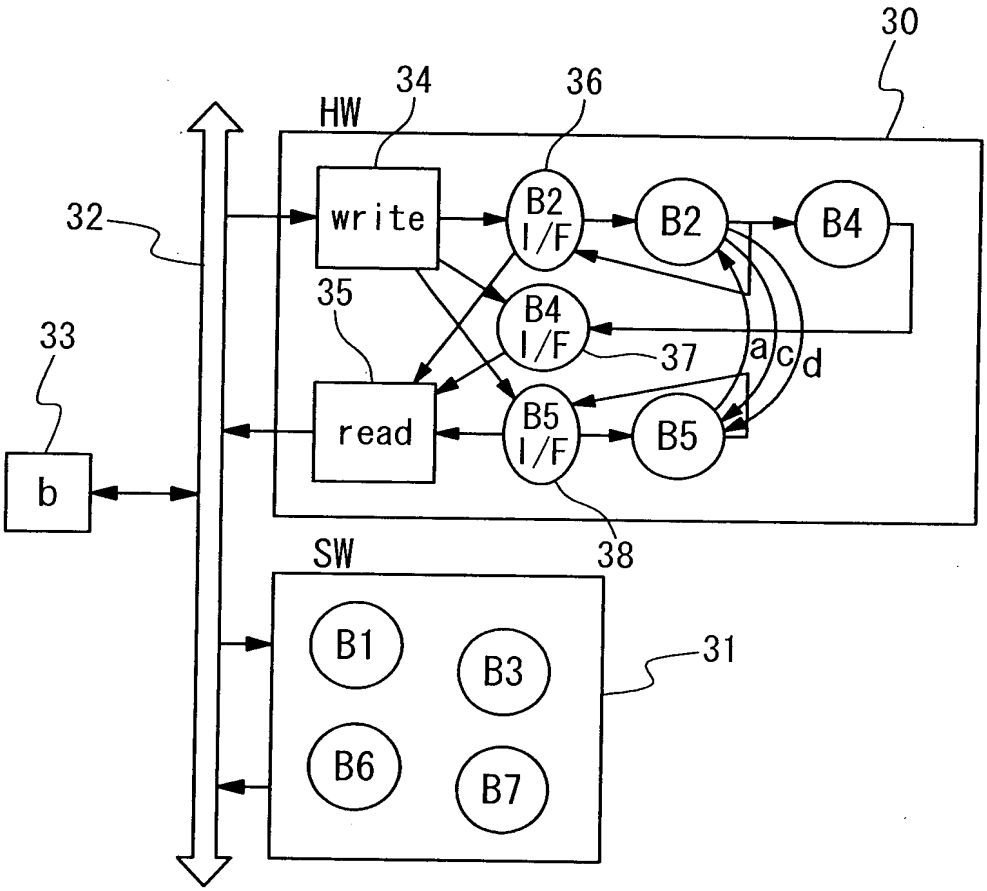

30
34
36
HW
32
write
B2 I/F
B2
B4
35
B4 I/F
37
33
a
c
d
read
B5 I/F
B5
b
38
SW
B1
B3
31
B6
B7

Fig. 5

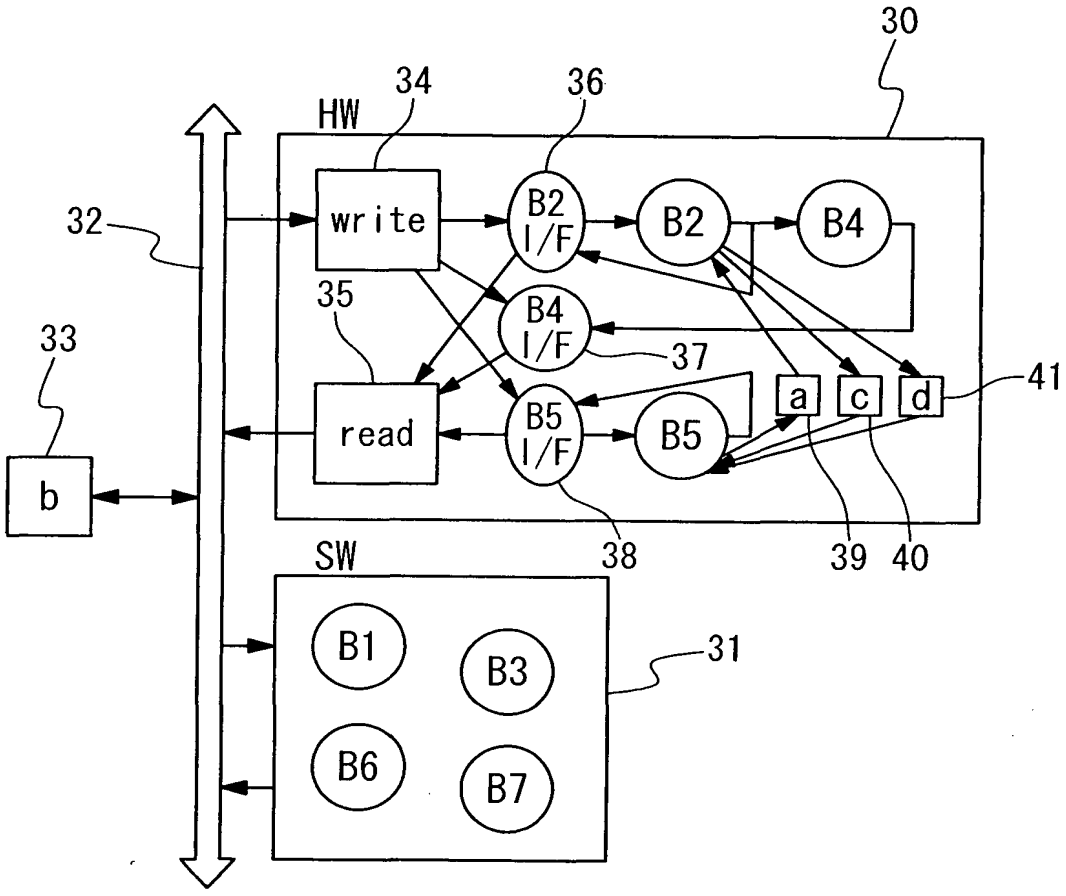

30
34
36
HW
32
write
B2 I/F
B2
B4
35
B4 I/F
37
33
read
B5 I/F
B5
a
c
d
41
b
38
39
40
SW
B1
B3
31
B6
B7

Fig. 6

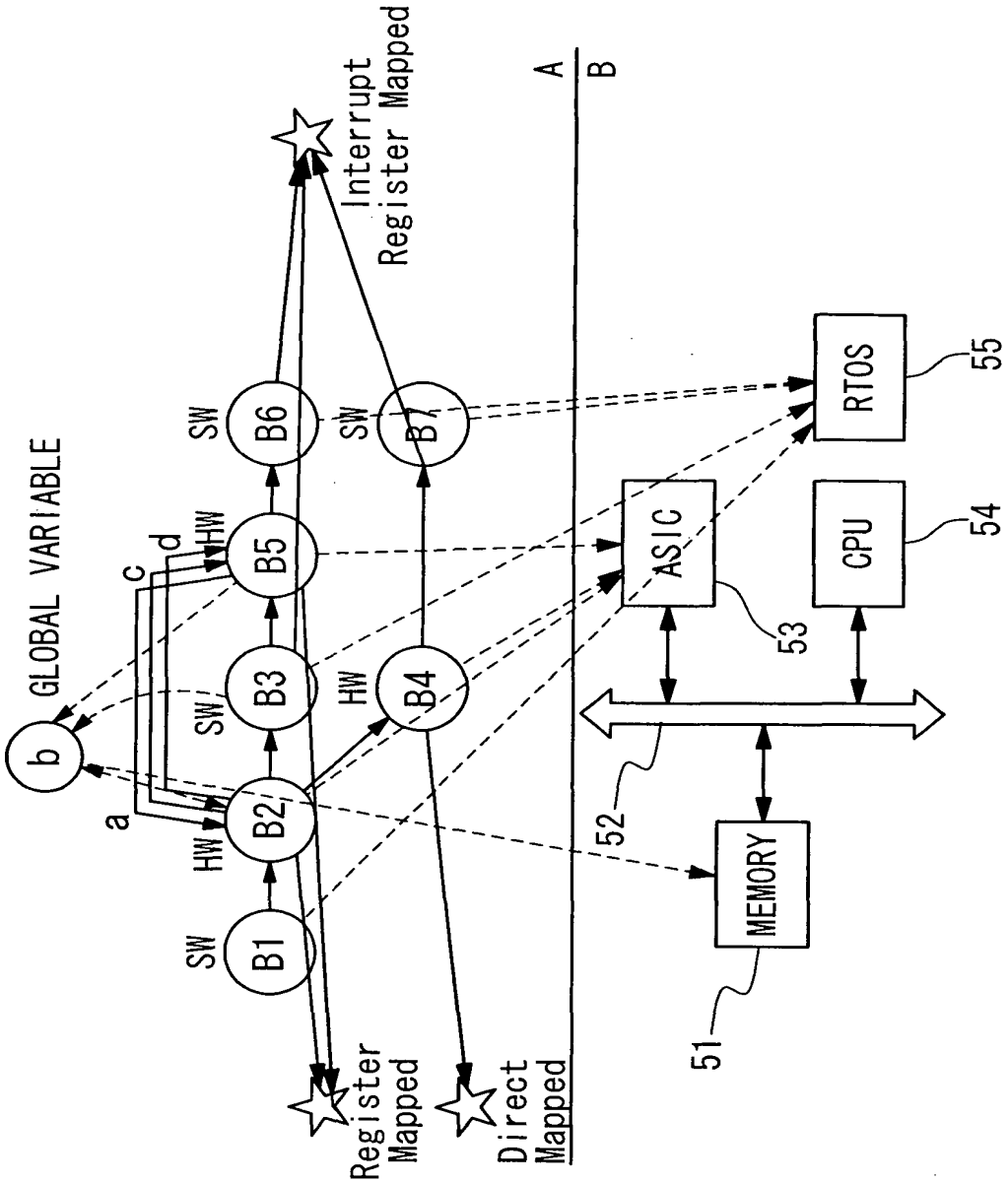

GLOBAL VARIABLE
b
a
c
d
SW
HW
B1
B2
B3
B4
B5
B6
B7
Register Mapped
Direct Mapped
Interrupt Register Mapped
A
B
RTOS
55
ASIC
CPU
54
53
52
MEMORY
51

Fig. 7

```
BUS bus_1("Bus1", arbitor_1, busread_1, buswrite_1);
Int arbitor_1(void);
RDATA busread_1(int add, int byte_count);
Int buswrite_1(int add, int data, int byte_count);

CPU ("cpu_1",&cpu_1_busif,0,0x10000,0x00ffe000,0x1000,0x8000);
BusIntf cpu_1_busif("cpu 1 Bus I/F",&bus_1,1)

SDRAM ExtSDRam1("External SDRam 1" , EXTSDRAM1_START_ADD,EXTSDRAM1_SIZE)

c_cpuifB2 cpuifB2(); c_cpuifB4 cpuifB4(); c_cpuifB5 cpuifB5();
c_B2 funcB2(); c_B4 funcB4(); c_B5 funcB5();

SystemoneScep()
{
  funcB2.OneStep(); cpuifB2.OneStep();
  funcB4.OneStep(); cpuifB4.OneStep();
  funcB5.OneStep(); cpuifB5.OneStep();
}
Systeminit()
{
  funcB2.init(); cpuifB2. int();
  funcB4.init(); cpuifB4. int();
  funcB5.init(); cpuifB5. int();
}
SystemReset()
{
  funcB2.Reset(); cpuifB2. Reset();
  funcB4.Reset(); cpuifB4. Reset();
  funcB5.Reset(); cpuifB5. Reset();
}
```

Fig. 8

```
RDATA busread_1 (int add, int byte_count)  {
  if ((add < start_bus_address) || (add > end_bus_address))  {
    return(NULL);
  }
  //RAM
  if ((add >= ExternalRAM.StartAddress) && (add < ExternalRAM. EndAddress))  {
    return ExternalRAM.Read(add - ExternalRAM.StartAddress, byte_count);
  }

  //ASIC
  if ((start_sub_address+B2_start_offset <= add)
       && (add < start_sub_address+B2_end_offset))  {
    cpuifB2. cpuread = 0;
    cpuifB2. bus_addin = add;
    rd.Status = cpuifB2.ReadIOReg();
    rd.data = cpuifB2.dataout;
   }
  if ((start_sub_address+B4_start_offset <= add)
       && (add < start_sub address+B4_end_offset))  {
    cpuifB4. cpuread = 0;
    cpuifB4. bus_addin = add;
    rd.Status = cpuifB4 ReadIOReg();
    rd.data = cpuifB4.dataout;
  }
  if ((start_sub_address+B5_start_offset <= add)
       && (add < start_sub_address+B5_ens_offset))  {
    cpuifB5. cpuread = 0;
    cpuifB5. bus_addin = add;
    rd.Status = cpuifB5.ReadIOReg();
    rd.data = cpuifB5.dataout;
  }

  ---------
  return (rd);
}
```

Fig. 9

```
RDATA buswrite_1 (int add, int byte_count)  {
  if ((add < start_bus_address) || (add > end_bus_address))  {
    return(1);
  }
  //RAM
  if ((add >= ExternalRAM.StartAddress) && (add < ExternalRAM. EndAddress))  {
    return ExternalRAM.Write(add - ExternalRAM.StartAddress, byte_count);
  }

  //ASIC
  if ((start_sub_address+B2_start_offset <= add)
      && (add < start_sub_address+B2_end_offset))  {
    cpuifB2. cpuwrite = 1;
    cpuifB2. bus_addin = add;
    cpuifB2. datain = data;
    rd.Status = cpuifB2.WriteIOReg();
   }
  if ((start_sub_address+B4_start_offset <= add)
      && (add < start_sub address+B4_end_offset))  {
    cpuifB4. cpuwrite = 1;
    cpuifB4. bus_addin = add;
    cpuifB4. datain = data;
    rd.Status = cpuifB4.WriteIOReg();
  }
  if ((start_sub_address+B5_start_offset <= add)
      && (add < start_sub_address+B5_ens_offset))  {
    cpuifB5. cpuwrite = 1;
    cpuifB5. bus_addin = add;
    cpuifB5. datain = data;
    rd.Status = cpuifB5.WriteIOReg();
  }

  ---------
  return (0);
}
```

Fig. 10

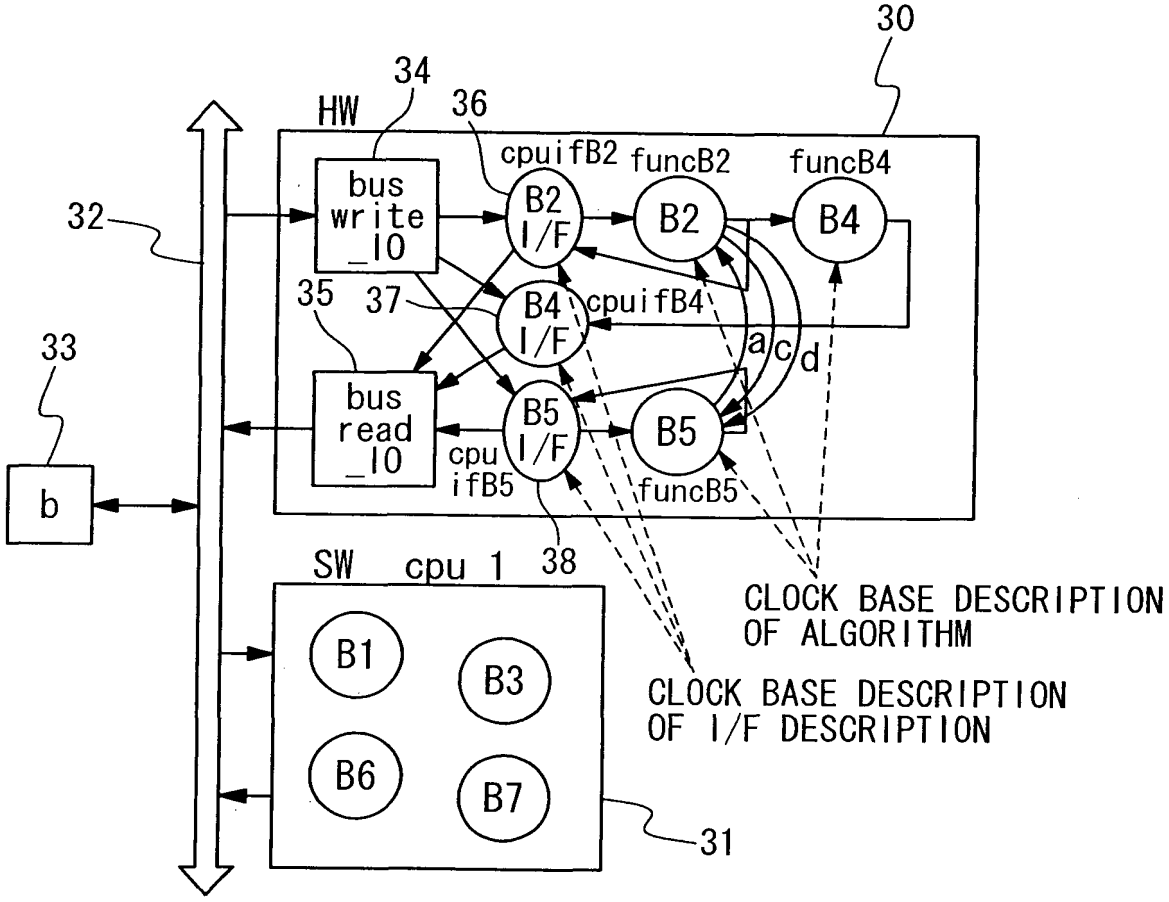

30
34
HW
36
cpuifB2
funcB2
funcB4
32
bus write _IO
B2 I/F
B2
B4
35
37
B4 I/F
cpuifB4
a
c
d
33
bus read _IO
B5 I/F
B5
b
cpu ifB5
funcB5
SW cpu 1
38
CLOCK BASE DESCRIPTION OF ALGORITHM
CLOCK BASE DESCRIPTION OF I/F DESCRIPTION
B1
B3
B6
B7
31

Fig. 11

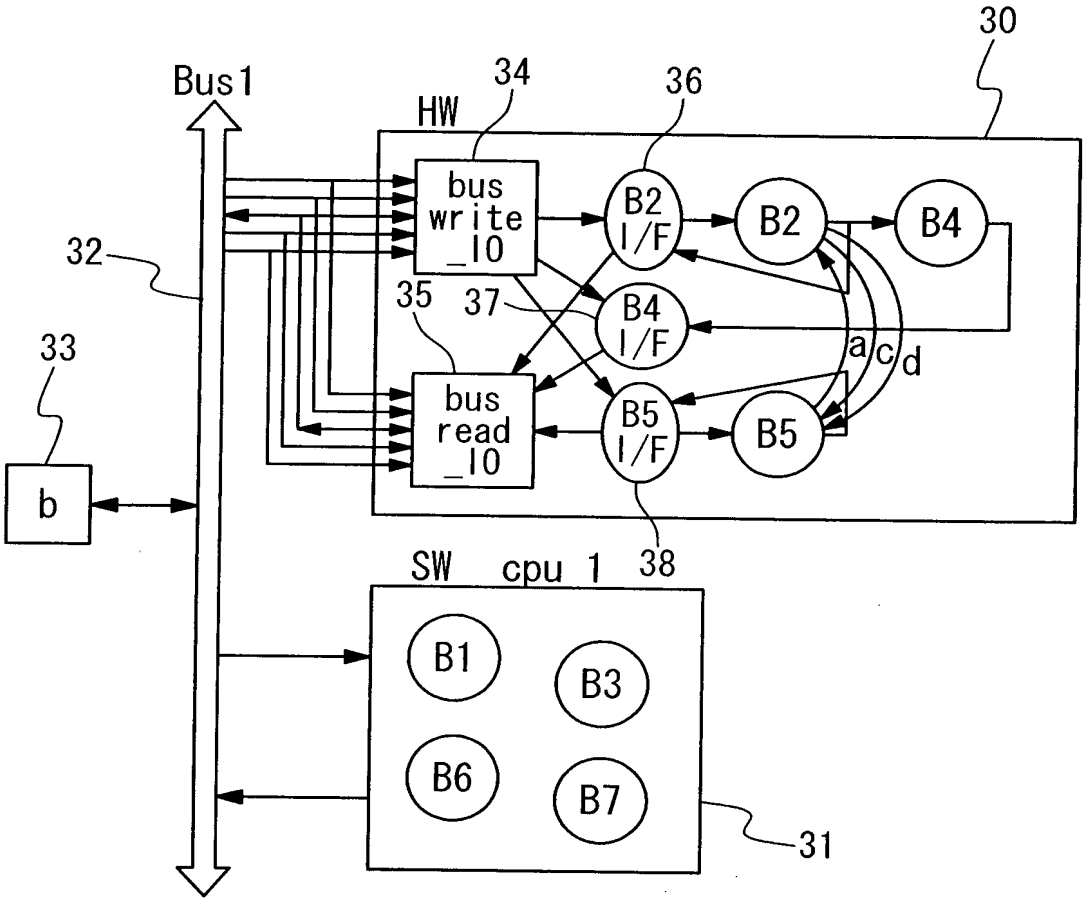

30
Bus1
34
36
HW
bus write _IO
B2 I/F
B2
B4
32
35
37
B4 I/F
33
a
c
d
bus read _IO
B5 I/F
B5
b
38
SW cpu 1
B1
B3
B6
B7
31

Fig. 12

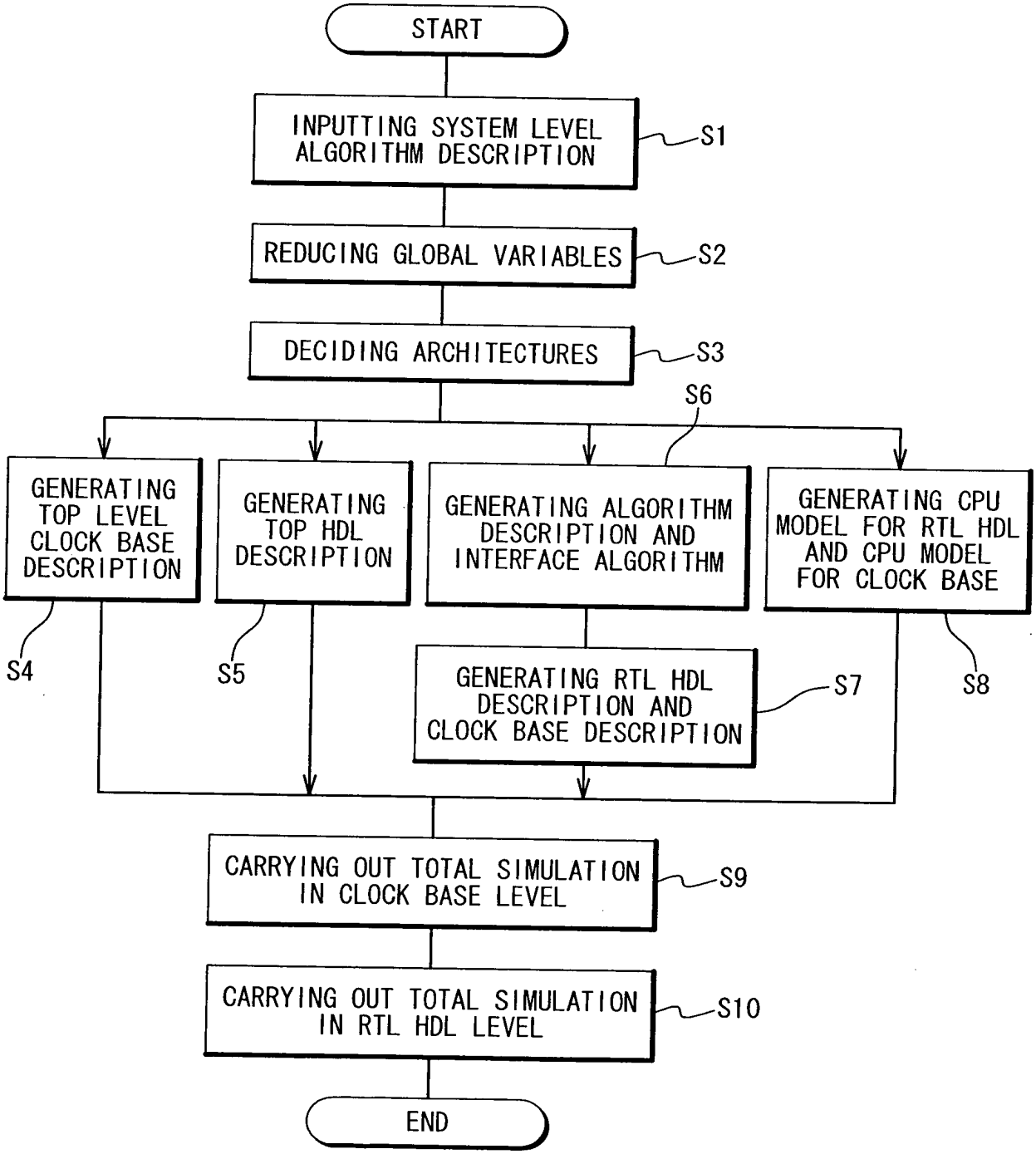

START
INPUTTING SYSTEM LEVEL ALGORITHM DESCRIPTION
S1
REDUCING GLOBAL VARIABLES
S2
DECIDING ARCHITECTURES
S3
S6
GENERATING TOP LEVEL CLOCK BASE DESCRIPTION
GENERATING TOP HDL DESCRIPTION
GENERATING ALGORITHM DESCRIPTION AND INTERFACE ALGORITHM
GENERATING CPU MODEL FOR RTL HDL AND CPU MODEL FOR CLOCK BASE
S4
S5
GENERATING RTL HDL DESCRIPTION AND CLOCK BASE DESCRIPTION
S7
S8
CARRYING OUT TOTAL SIMULATION IN CLOCK BASE LEVEL
S9
CARRYING OUT TOTAL SIMULATION IN RTL HDL LEVEL
S10
END

Fig. 13

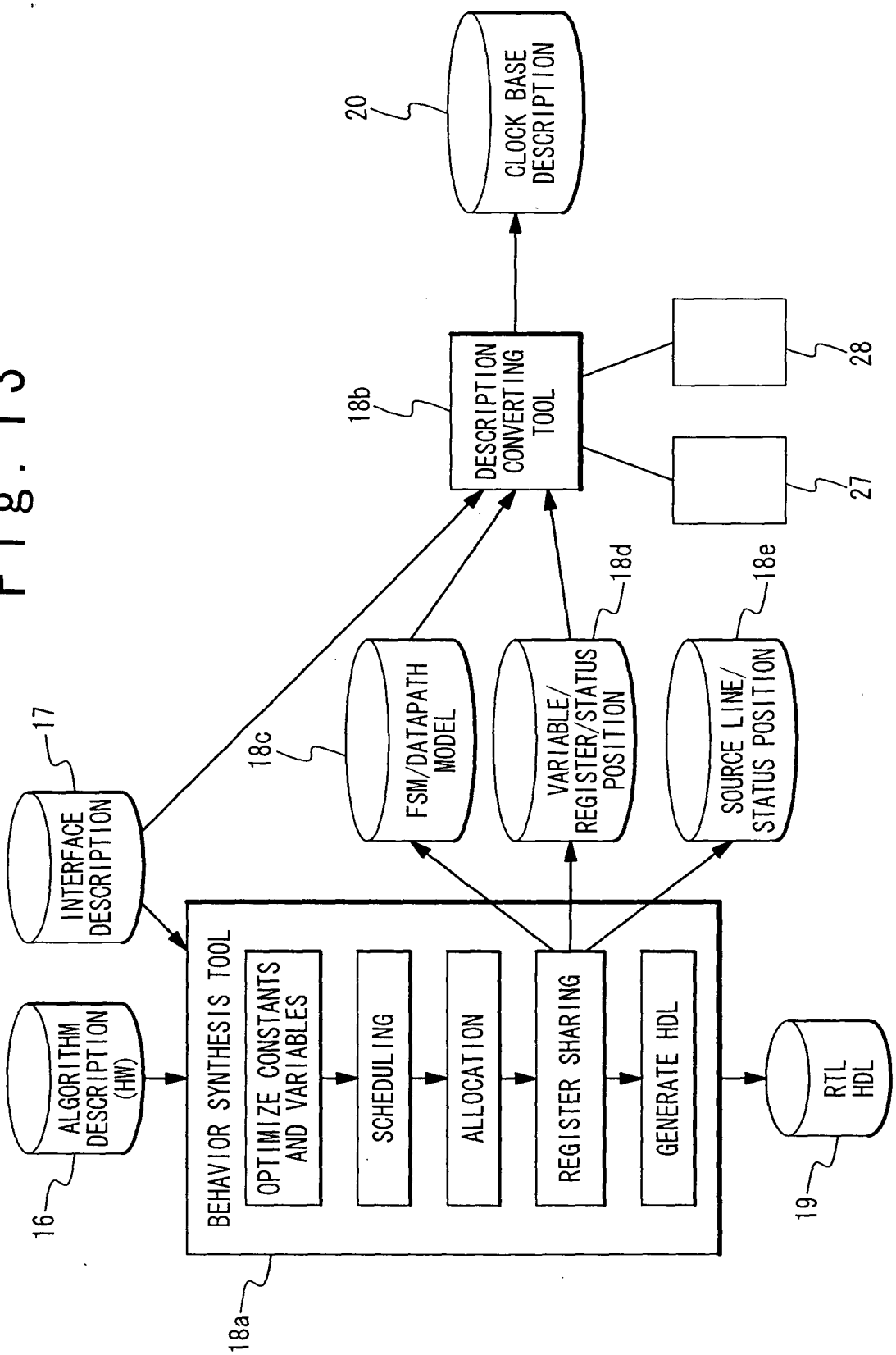
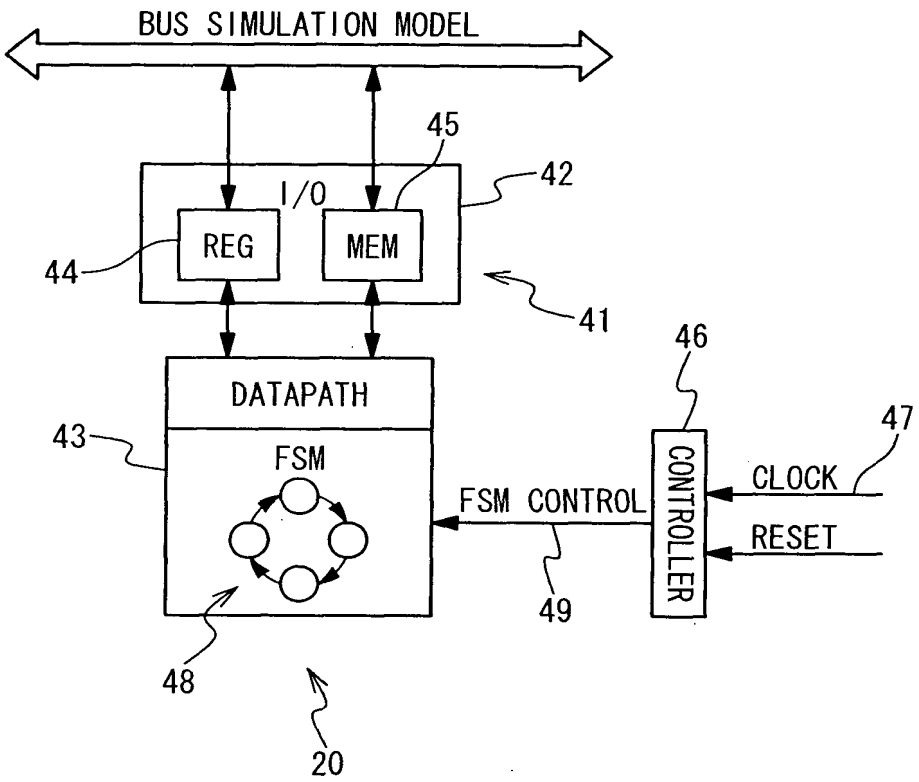

ALGORITHM DESCRIPTION (HW)
16
INTERFACE DESCRIPTION
17
BEHAVIOR SYNTHESIS TOOL
OPTIMIZE CONSTANTS AND VARIABLES
SCHEDULING
ALLOCATION
REGISTER SHARING
GENERATE HDL
18a
RTL HDL
19
FSM/DATAPATH MODEL
18c
VARIABLE/ REGISTER/STATUS POSITION
18d
SOURCE LINE/ STATUS POSITION
18e
DESCRIPTION CONVERTING TOOL
18b
CLOCK BASE DESCRIPTION
20
27
28

Fig. 14


BUS SIMULATION MODEL
45
42
I/O
REG
MEM
44
41
46
DATAPATH
47
43
FSM
CLOCK
FSM CONTROL
CONTROLLER
RESET
49
48
20